Computer Communications 144 (2019) 18–30

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Machine learning for measurement-based bandwidth estimation ☆

Sukhpreet Kaur Khangura *, Markus Fidler, Bodo Rosenhahn

Department of Electrical Engineering and Computer Science, Leibniz University Hannover, Germany

ARTICLE INFO

Keywords:
Available bandwidth estimation
Machine learning

ABSTRACT

The dispersion that arises when packets traverse a network carries information that can reveal relevant network characteristics. Using a fluid-flow model of a bottleneck link with first-in first-out multiplexing, accepted probing tools measure the packet dispersion to estimate the available bandwidth, i.e., the residual capacity that is left over by other traffic. Difficulties arise, however, if the dispersion is distorted compared to the model, e.g., by non-fluid traffic, multiple bottlenecks, clustering of packets due to interrupt coalescing, and inaccurate time-stamping in general. It is recognized that modeling these effects is cumbersome if not intractable. This motivates us to explore the use of machine learning in bandwidth estimation. We train a neural network using vectors of the packet dispersion that is characteristic of the available bandwidth. Our testing results reveal that even a shallow neural network identifies the available bandwidth with high precision. We also apply the neural network under a variety of notoriously difficult conditions that have not been included in the training, such as randomly generated networks with the multiple bottleneck links and heavy cross traffic burstiness. Compared to two state-of-the-art model-based techniques as well as a recent machine learning-based technique (Yin et al., 2016), our neural network approach shows improved performance. Further, our neural network can effectively control the estimation procedure in an iterative implementation. We also evaluate our method with other supervised machine learning techniques.

1. Introduction

The term *available bandwidth* refers to the residual capacity of a link or a network path that is left over after the existing traffic, also referred to as *cross traffic*, is served. Knowledge of the available bandwidth benefits rate-adaptive applications and facilitates, e.g., network monitoring, detection of congested links, and load balancing. The goal of bandwidth estimation is to infer the available bandwidth of a network path using external observations of data packets, only.

Formally, given a link with capacity C and cross traffic with long-term average rate λ, where $\lambda \in [0, C]$, the available bandwidth $A \in [0, C]$ is defined as $A = C - \lambda$ [1]. The end-to-end available bandwidth of a network path is determined by its *tight link*, that is the link that has the minimal available bandwidth [2]. The tight link may differ from the *bottleneck link*, i.e., the link with the minimal capacity.

To date, a number of accepted active probing techniques and corresponding theories for available bandwidth estimation exist, e.g., [1–13]. These techniques use a sender that actively injects artificial *probe traffic* with a defined packet size l and inter packet gap referred to as input gap g_{in} into the network. At the receiver, the output gap of the received probe g_{out} is measured to deduce the available bandwidth.

A common assumption in bandwidth estimation is that the available bandwidth, respectively, the rate of the cross traffic does not change during a probe. Further, to simplify modeling, cross traffic is assumed to behave like fluid, i.e., effects that are due to the packet granularity of the cross traffic are neglected. Modeling a single tight link as a lossless First-In First-Out (FIFO) multiplexer of probe and cross traffic, the relation of g_{out} and g_{in} follows by an intuitive argument [1] as

$$g_{\text{out}} = \max\left\{ g_{\text{in}}, \frac{g_{\text{in}}\lambda + l}{C} \right\}. \tag{1}$$

The reasoning is that during g_{in} an amount of $g_{\text{in}}\lambda$ of the fluid cross traffic is inserted between any two packets of the probe traffic, so that the probe packets may be spaced further apart. Reordering Eq. (1) gives the characteristic *gap response curve*

$$\frac{g_{\text{out}}}{g_{\text{in}}} = \begin{cases} 1 & \text{if } \frac{l}{g_{\text{in}}} \le C - \lambda, \\ \frac{l}{g_{\text{in}} C} + \frac{\lambda}{C} & \text{if } \frac{l}{g_{\text{in}}} > C - \lambda. \end{cases} \tag{2}$$

The utility of Eq. (2) is that it shows a clear bend at $A = C - \lambda$ that enables estimating the available bandwidth using different techniques,

☆ This manuscript is a revised and extended version of the paper Khangura et al. (2018) that appeared in the IFIP Networking 2018 proceedings.

* Corresponding author.
E-mail addresses: khangura.sukhpreet@ikt.uni-hannover.de (S.K. Khangura), markus.fidler@ikt.uni-hannover.de (M. Fidler), rosenhahn@tnt.uni-hannover.de (B. Rosenhahn).

https://doi.org/10.1016/j.comcom.2019.05.005

Available online 17 May 2019

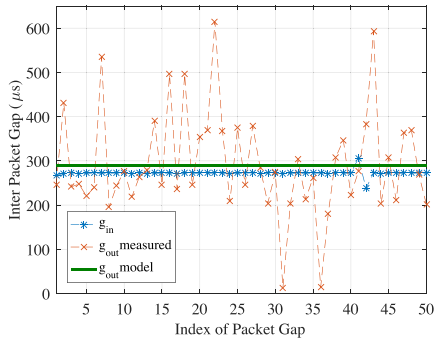


Fig. 1. Measurements of g_{in} and g_{out} compared to the fluid model. The network has a single tight link with capacity $C = 100$ Mbps and exponential cross traffic with rate $\lambda = 62.5$ Mbps. The packet size is $l = 1514$ byte.

see Section 2. We note that the quotient of packet size and gap is frequently viewed as the data rate of the probe.

For an example, consider a tight link with capacity $C = 100$ Mbps and cross traffic with average rate $\lambda = 62.5$ Mbps. The packet size is $l = 1514$ byte, resulting in a transmission time l/C of about 120 μs. Given an input gap $g_{in} = 270$ μs, the output gap follows from Eq. (1) as $g_{out} = 290$ μs. Now, assume for a moment that C is known but λ is unknown. An active probing tool can send probes with, e.g., $g_{in} = 270$ μs to measure g_{out}. Noting that $g_{out}/g_{in} > 1$, Eq. (2) reveals the unknown $\lambda = (g_{out}C - l)/g_{in} = 62.5$ Mbps.

In practice, the observations of g_{out} are distorted for various reasons. For an example, we recorded a measurement trace of 50 pairs of g_{in} and g_{out} in the network testbed in Fig. 3 with a single tight link and the above parameters C, λ, and l, where l is the maximal size of Ethernet packets including the header. The results are shown in Fig. 1. Neglecting the cases where $g_{out} < g_{in}$ that are not possible in the model and ignoring large outliers, a range of samples g_{out} of about 360 μs remain that suggest concluding $\lambda \approx 90$ instead of 62.5 Mbps erroneously.

1.1. Challenges in bandwidth estimation

Relevant reasons for the distortions of g_{out} include deviations from the assumptions of the model, i.e., a lossless FIFO multiplexer with constant, fluid cross traffic as well as measurement inaccuracies, such as imprecise time-stamping:

Random cross traffic. Eq. (2) is deterministic and hence it does not define how to deal with the randomness of g_{out} that is caused by variable bit rate cross traffic. It is shown in [1] that the problem cannot be easily fixed by using the expected value $\mathsf{E}[g_{out}]$ instead. In brief, this is due to the non-linearity of Eq. (2) and the fact that the location of the turning point $C - \lambda$ fluctuates if the rate of the cross traffic λ is variable. The result is a deviation that is maximal at $l/g_{in} = C - \lambda$ and causes underestimation of the available bandwidth [1,14].

Packet interference. Non-conformance with the fluid model arises due to the interaction of probes with packets of the cross traffic. In Fig. 1, two relevant examples are identified by frequent samples of g_{out} in the range of 240 and 360 μs, respectively. In contrast, the g_{out} of 290 μs, that is predicted by the fluid model, is observed rarely. To understand this effect, consider two probe packets with $g_{in} = 270$ μs and note that the transmission time of a packet is 120 μs. The case $g_{out} = 360$ μs occurs if two cross traffic packets are inserted between the two probe packets. Instead, if one of the two cross traffic packets is inserted in front of the probe, it delays the first probe packet, resulting in $g_{out} = 240$ μs.

Packet loss. If probe packets are lost, the corresponding output gaps are void. This causes estimation bias, since packets that encounter congestion have a higher loss probability. There are few bandwidth estimation tools that consider loss, e.g., as an indication that the available bandwidth is exceeded [6].

Multiple tight links. An extension of Eq. (1) for multiple links is derived in [3]. Yet, if cross traffic is non-fluid, the repeated packet interaction at each of the links distorts the probe gaps. Further, in case of random cross traffic, there may not be a single tight link, but the tight link may vary randomly. The consequence is an underestimation of the available bandwidth [14,15] that is analyzed in [11,13].

Measurement inaccuracies. Besides, there exist limitations of the accuracy due to the hardware of the hosts where measurements are taken. A possible clock offset between sender and receiver is dealt with by the use of probe gaps. A problem in high-speed networks is, however, interrupt coalescing [16,17]. This technique avoids flooding a host with too many interrupts by grouping packets received in a short time together in a single interrupt, which distorts g_{out}.

1.2. State-of-the-art estimation techniques

To alleviate the observed variability of the samples of g_{out}, state-of-the-art bandwidth estimation methods perform averaging of several g_{out} samples. These samples can be collected by repeated probes of two packets, so-called *packet pairs* [18], or by *packet trains* [5,19] that consist of n consecutive packets and hence comprise $n - 1$ gaps. Further, to improve the available bandwidth estimates, statistical post-processing techniques are used, such as Kalman filtering [10,20], majority decisions [6], averaging of the bandwidth estimates of repeated experiments [7,8], or linear regression [4].

These techniques do, however, not overcome the basic assumptions of the deterministic fluid model in Eq. (1). While packet trains and statistical postprocessing help reduce the variability of available bandwidth estimates, these cannot resolve systematic deviations, such as the underestimation bias in case of random cross traffic and multiple tight links [1,11,13]. Further, it is difficult to tailor methods to specific hardware implementations that influence the measurement accuracy.

These fundamental limitations motivate us to explore the use of machine learning in available bandwidth estimation. The machine learning approach has been considered earlier in [21–23] and receives increasing attention in the recent research [17,24]. The works differ from each other with respect to their application: [21] considers the prediction of the available bandwidth from packet data traces that have been obtained in passive measurements. In contrast [17,22–24] use active probes to estimate the available bandwidth in NS-2 simulations [22], the ETOMIC network infrastructure [23], ultra-high speed 10 Gbps networks [17], and operational LTE networks [24], respectively.

Common to these active probing methods [17,22–24] is the use of packet chirps [7] that are probes of several packets sent at an increasing data rate. The rate increase is achieved either by a (geometric) reduction of the input gap [22,23], by concatenating several packet trains with increasing rates to a multi-rate probe [17], or by a linear increase of the packet size [24]. Chirps permit detecting the turning point of Eq. (2), that coincides with the available bandwidth, using a single probe. They are, however, susceptible to random fluctuations [12].

Other than chirps, [22] evaluates packet bursts that are probes of back-to-back packets and concludes that bursts are not adequate to estimate the available bandwidth. Also, [17] considers constant rate packet trains for an iterative search for the available bandwidth. Here, machine learning solves a classification problem to estimate whether the rate of a packet train exceeds the available bandwidth or not. Depending on the result, the rate of the next packet train is reduced or increased in a binary search as in [6] until the probe rate approaches the available bandwidth. The authors of [17] give,

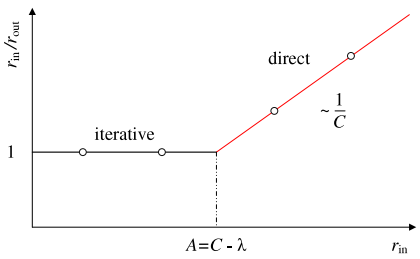


Fig. 2. Rate response curve. The turning point marks the available bandwidth.

however, preference to chirp probes. The feature vectors that are used for machine learning are generally measurements of g_{out} [22–24] with the exception that [17] uses the Fourier transform of vectors of g_{in} and g_{out}. Supervised learning is used and [17,24] take advantage of today's availability of different software packages to compare the utility of state-of-the-art machine learning techniques in bandwidth estimation.

1.3. Contributions

In this work, we investigate how to benefit from machine learning, specifically neural networks, when using standard packet train probes for available bandwidth estimation. Compared to packet chirps, that are favored in the related works [17,22,24], packet trains have been reported to be more robust to random fluctuations. In fact, the implementation of a chirp as a multi-rate probe [17], that concatenates several packet trains with increasing rates, also benefits from this.

Different from multi-rate probes, packet trains are typically used in an iterative procedure that takes advantage of feedback to adapt the rate of the next packet train. Such a procedure is also proposed in [17], where machine learning is used to classify individual packet trains to control a binary search. The goal is to adapt the probe rate until it approaches the available bandwidth. In contrast, we use a feature vector that iteratively includes each additional packet train probe. This additional information enables estimating the available bandwidth directly, without the necessity that the probe rate converges to the available bandwidth. Instead of a binary search, our method chooses the probe rate next, that is expected to improve the bandwidth estimate the most.

We evaluate our method in controlled experiments in a network testbed. We specifically target topologies where the assumptions of the deterministic fluid model in Eq. (1) are not satisfied, such as bursty cross traffic, and multiple tight links. For a reference, we implement three state-of-the-art methods, two model-based and one machine learning-based, to use the same data set as our neural network-based approach.

This work is an extended version of our earlier paper [25]. Compared to [25], we present a broader evaluation of our method using a wide range of randomly generated topologies with largely varying capacities and cross-traffic intensities and burstiness. The results confirm that our method is scale-invariant, i.e., it is applicable without prior calibration to networks of arbitrary capacity. We also train our method for multiple tight links to reduce the bias and variance in estimates. Further, we present a deeper investigation of multi-hop networks. We specifically consider cases where the tight link differs from the bottleneck link and show how the order in which these occur in a network path affects the input–output relation (1) that is used for available bandwidth estimation. To compare with state-of-the-art machine learning technique [17], we formulate the task of available bandwidth estimation as a classification problem. We implement two different classifiers named individual classifier and full classifier. The individual classifier represents state-of-the-art and uses information of a single probe rate to classify whether the current probe rate is above or below the available bandwidth. Depending upon the result, the next probe rate is chosen iteratively until the probe rate converges to the available bandwidth. In contrast, the full classifier that is implemented by our method uses a k-dimensional feature vector which includes the information of all the previous probe rates to improve the classification whether the probe rates are above or below the available bandwidth.

Further, we evaluate our proposed method with three other supervised machine learning techniques: Support Vector Regression, Gaussian Process Regression, and Random Forest. We show that the ability of the neural network to generalize non-locally makes them favorable for the available bandwidth estimation in arbitrary topologies where the network parameters differ significantly from the training data. We provide data sets generated in a controlled network testbed located at Leibniz University Hannover for training and testing of our proposed method.

The remainder of this paper is structured as follows: In Section 2, we introduce the reference implementation of model-based estimation techniques. We present our neural network-based method, describe the training, and show testing results in Section 3. In Section 4, we consider the estimation of available bandwidth and capacity for randomly generated networks with different tight link capacities and networks with multiple tight links. Our iterative neural network-based method that selects the probe rates itself as well as a comparison with other machine learning techniques are presented in Section 5. In Section 6, we give brief conclusions. In Section 7, we provide overview of the data sets used for training and evaluation.

2. Model-based reference implementations

The methods for available bandwidth estimation that are based on the fluid model of Eq. (1) essentially fall into two different categories: *iterative probing* and *direct probing*. For each of the two categories, we implement a bandwidth estimation technique that is representative of state-of-the-art. While available bandwidth estimation tools differ significantly regarding the selection and the amount of probe traffic, our implementations are tailored to use the same database so that they provide a reference for the neural network-based method.

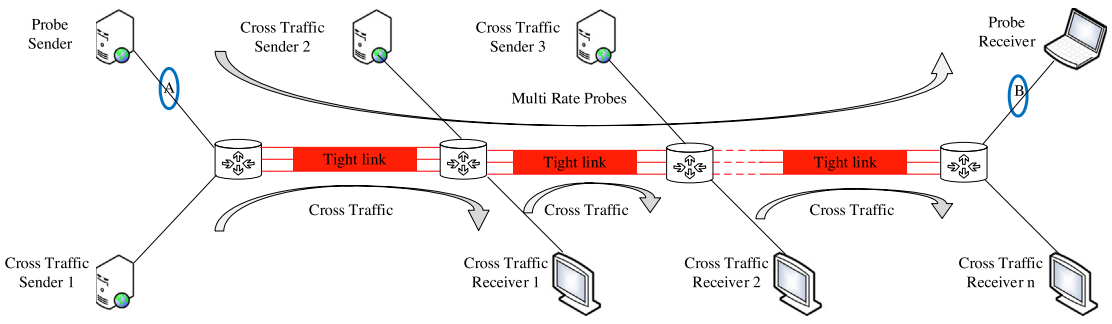


Fig. 3. Dumbbell topology set up using the Emulab and MoonGen software. A varying number of tight links with single hop-persistent cross traffic are configured. Probe-traffic is path-persistent to estimate the end-to-end available bandwidth from measurements at points A and B.

To reduce the variability of the measurements, a common approach is the use of constant rate packet train probes. A packet train of n packets comprises $n-1$ gaps. At the receiver, the gaps are defined as $g_{\text{out}}(j) = t_{\text{out}}(j+1) - t_{\text{out}}(j)$ for $j = 1 \dots n-1$, where $t_{\text{out}}(j)$ is the receive time-stamp of packet j. Considering the output rate of a packet train defined as

$$r_{\text{out}} = \frac{(n-1)l}{t_{\text{out}}(n) - t_{\text{out}}(1)} \tag{3}$$

implies averaging of the output gaps since by definition of $g_{\text{out}}(j)$, Eq. (3) can be rewritten as

$$r_{\text{out}} = \frac{l}{\frac{1}{n-1}\sum_{j=1}^{n-1} g_{\text{out}}(j)}.$$

In case of long packet trains, stationarity, and ergodicity, the denominator converges to the mean $\overline{g_{\text{out}}}$. Further, for the deterministic fluid model, $g_{\text{out}}(j) = g_{\text{out}}$ for $j = 1 \dots n-1$ so that $r_{\text{out}} = l/g_{\text{out}}$. Similarly, the input rate for a defined g_{in} is $r_{\text{in}} = l/g_{\text{in}}$ and by insertion into Eq. (2) the equivalent *rate response curve* of the fluid model is obtained as

$$\frac{r_{\text{in}}}{r_{\text{out}}} = \begin{cases} 1 & \text{if } r_{\text{in}} \le C - \lambda, \\ \frac{r_{\text{in}}+\lambda}{C} & \text{if } r_{\text{in}} > C - \lambda. \end{cases} \tag{4}$$

The characteristic shape of Eq. (4) is shown in Fig. 2.

2.1. Iterative probing

In brief, iterative probing techniques search for the turning point of the rate response curve by sending repeated probes at increasing rates, as long as $r_{\text{in}} = r_{\text{out}}$. When r_{in} reaches $C - \lambda$, the available bandwidth is saturated and increasing the probe rate r_{in} further results in self-induced congestion, so that $r_{\text{in}} > r_{\text{out}}$. This implies queueing at the tight link and hence increasing one way delays can be observed at the receiver.

Established iterative probing tools are, e.g., Pathload [6] and IGI/PTR [9]. Pathload adaptively varies the rates of successive packet trains r_{in} in a binary search until r_{in} converges to the available bandwidth. It uses feedback from the receiver that reports whether r_{in} exceeds the available bandwidth or not. The decision is made based on two statistical tests that detect increasing trends of the one way delay. For comparison, IGI/PTR tests whether $(r_{\text{in}} - r_{\text{out}})/r_{\text{in}} > \Delta\text{th}$, where the threshold value Δth is set to 0.1, to detect whether the probe rate exceeds the available bandwidth. Regarding the variability of the available bandwidth, Pathload reports an available bandwidth range that is determined by the largest probe rate that did not cause self-induced congestion and the smallest rate that did cause congestion, respectively.

In our experiments, we use a data set of equidistantly spaced r_{in} and corresponding r_{out}. We process these entries iteratively in increasing order of r_{in} and apply the threshold test of IGI/PTR [9] $(r_{\text{in}} - r_{\text{out}})/r_{\text{in}} > \Delta\text{th}$ to determine whether r_{in} exceeds the available bandwidth. We denote r_{in}^{th} the largest rate before the test detects that the available bandwidth is exceeded for the first time and report r_{in}^{th} as the available bandwidth estimate. We note that there may, however, exist $r_{\text{in}} > r_{\text{in}}^{th}$, where the test fails again. This may occur, for example, due to the burstiness of the cross traffic that causes fluctuations of the available bandwidth.

2.2. Direct probing

Instead of searching for the turning point of the rate response curve, direct probing techniques seek to estimate the parameters of the upward line segment for $r_{\text{in}} > C - \lambda$. The line is determined by C and λ. If C is known, a single probe $r_{\text{in}} = C$ yields a measurement of r_{out} that is sufficient to estimate $\lambda = C(C/r_{\text{out}} - 1)$ from Eq. (4). Spruce [8] implements this approach. If C is also unknown, at least two different probe rates $r_{\text{in}} > C - \lambda$ are needed to estimate the two unknown parameters of the upward line segment of the rate response curve. This approach is taken, e.g., by TOPP [3], DietTOPP [4], and BART [10].

To implement the direct probing technique, we combine it with a threshold test to select relevant probe rates. Direct probing techniques require that $r_{\text{in}} > C - \lambda$ where C and λ are unknown. We adapt a criterion from DietTOPP [4] to determine a minimum threshold $r_{\text{in}}^{\min}$ that satisfies $r_{\text{in}}^{\min} > C - \lambda$ and use only the probe rates $r_{\text{in}} \ge r_{\text{in}}^{\min}$.

The procedure to find $r_{\text{in}}^{\min}$ is as follows: We use the maximal input rate in the measurement data denoted by $r_{\text{in}}^{\max}$ and extract the corresponding output rate $r_{\text{out}}^{\max}$. If $r_{\text{in}}^{\max} > r_{\text{out}}^{\max}$, it can be seen from Eq. (4) that both $r_{\text{in}}^{\max} > C - \lambda$ as well as $r_{\text{out}}^{\max} > C - \lambda$. Hence, we use $r_{\text{in}}^{\min} = r_{\text{out}}^{\max}$ as a threshold to filter out all $r_{\text{in}} \le r_{\text{in}}^{\min}$. Once we have selected samples that certainly fulfill $r_{\text{in}} > C - \lambda$, we use linear regression like [3,4] to determine the upward segment of the rate response curve. The available bandwidth estimate is determined from Eq. (4) as the x-axis intercept where the regression line intersects with the horizontal line at 1, see Fig. 2.

If the assumptions of the fluid model do not hold, e.g., in case of random cross traffic, the regression technique may occasionally fail. We filter out bandwidth estimates that can be classified as infeasible. This is the case if the slope of the regression line is so small that the intersection with 1 is on the negative r_{in} axis, implying the contradiction $A < 0$, or if the slope of the regression line is negative, implying $C < 0$.

3. Neural network-based method

In this section, we present our neural network-based implementation of bandwidth estimation, describe the training data sets, and show a comparison of available bandwidth estimates for a range of different network parameters.

3.1. Scale-invariant implementation

We use a neural network that takes a k-dimensional vector of values $r_{\text{in}}/r_{\text{out}}$ as input. The corresponding r_{in} are equidistantly spaced with an increment δ_r. Hence, r_{in} is in $[\delta_r, 2\delta_r, \dots, k\delta_r]$ that is fully defined by the parameters k and δ_r that determine the measurement resolution. Since the actual values of r_{in} do not provide additional information, they are not input to the neural network. Instead, the neural network refers to values of $r_{\text{in}}/r_{\text{out}}$ only by their index $i \in [1, k]$. The output of the neural network is the tuple of bottleneck capacity and available bandwidth that are also normalized with respect to δ_r, i.e., we use C/δ_r and A/δ_r, respectively. While C/δ_r and A/δ_r are not necessarily integer, they can be thought of as the index i_C and i_A where r_{in} saturates the bottleneck capacity or the available bandwidth, respectively. To obtain the actual capacity and the available bandwidth, the output of the neural network has to be multiplied by δ_r.

Since the capacity of the network is a priori unknown, an adequate measurement resolution δ_r has to be estimated first. For this purpose, we use a packet train probe consisting of n packets that are sent as a burst, i.e., at the maximum possible rate denoted as r_{in}^{δ}, and measure the corresponding output rate r_{out}^{δ} at the receiver. Sending the packet train as a burst ensures that $r_{\text{in}}^{\delta} > C - \lambda$ so that we have $r_{\text{out}}^{\delta} = r_{\text{in}}^{\delta} C/(r_{\text{in}}^{\delta} + \lambda)$ from Eq. (4). Further, $r_{\text{out}}^{\delta} \in [C - \lambda, C]$ if $r_{\text{in}}^{\delta} \to C - \lambda$ or $r_{\text{in}}^{\delta} \to \infty$ respectively. Hence, r_{out}^{δ} provides a useful estimate of the probing range for bandwidth estimation. Since our feature vector is k-dimensional, we divide $\delta_r = r_{\text{out}}^{\delta}/k$ to determine the set of probing rates $r_{\text{in}} \in [\delta_r, 2\delta_r, \dots, k\delta_r]$. In practice, we use a smoothed estimate $\hat{r}_{\text{out}}^{\delta}$ that is obtained as the average output rate of a number of repeated probes.

The normalization by δ_r achieves a neural network that is scale-invariant, since the division by δ_r replaces the units, e.g., Mbps or Gbps, by indices. Considering the fluid model in Eq. (4), the normalization of all quantities r_{in}, r_{out}, C, and λ by δ_r results in

$$\frac{r_{\text{in}}}{r_{\text{out}}} = \begin{cases} 1 & \text{if } i \le i_C - i_\lambda, \\ \frac{i+i_\lambda}{i_C} & \text{if } i > i_C - i_\lambda. \end{cases} \tag{5}$$

where we used the indices $i = r_{\text{in}}/\delta_r$, $i_C = C/\delta_r$, $i_\lambda = \lambda/\delta_r$, and $i_A = A/\delta_r = i_C - i_\lambda$. Eq. (5) confirms that the shape of $r_{\text{in}}/r_{\text{out}}$ is independent of the scale, e.g., sampling a 100 Mbps network in increments of $\delta_r = 10$ Mbps or a 1 Gbps network in increments of $\delta_r = 0.1$ Gbps reveals the same characteristic shape. The advantage of the scale-invariant representation is that the neural network requires less training and is applicable to networks of arbitrary capacity. We note that the identity Eq. (5) is derived under the assumptions of the fluid model and does not consider effects that are not scale-invariant such as the impact of the packet size or interrupt coalescing.

For implementation we use a $k = 20$-dimensional input vector of equidistantly sampled values of $r_{\text{in}}/r_{\text{out}}$. We decided for a shallow neural network consisting of one hidden layer with 40 neurons. Thus, the network comprises a 20-dimensional input vector, 40 hidden neurons and two output neurons. The output neutrons encode C/δ_r and A/δ_r.

3.2. Training data: Exponential cross traffic, single tight link

We generate different data sets for training and evaluation using a controlled network testbed. The testbed is located at Leibniz Universität Hannover and comprises about 80 machines that are each connected by a minimum of 4 Ethernet links of 1 Gbps and 10 Gbps capacity via VLAN switches. The testbed is managed by the Emulab software [26] that configures the machines as hosts and routers and connects them using VLANs to implement the desired topology. We use a dumbbell topology with multiple tight links as shown in Fig. 3. To emulate the characteristics of the links, such as capacity, delay, and packet loss, additional machines are employed by Emulab. We use the MoonGen software [27] for emulation of link capacities that differ from the native Ethernet capacity. To achieve an accurate spacing of packets that matches the emulated capacity, MoonGen fills the gaps between packets by dummy frames that are discarded at the output of the link. We use the forward rate Lua script for the MoonGen API to achieve the desired forwarding rate for the transmission and reception ports of MoonGen.

Cross traffic of different types and intensities is generated using D-ITG [28]. The cross traffic is *single hop-persistent*, i.e., at each link fresh cross traffic is multiplexed. The probe traffic is *path-persistent*, i.e., it travels along the entire network path, to estimate the end-to-end available bandwidth. We use RUDE & CRUDE [29] to generate UDP probe streams. A probe stream consists of a series of k packet trains of n packets each. The k packet trains correspond to k different probe rates with a constant rate increment of δ_r between successive trains. The packet size of the probe traffic and the cross traffic is $l = 1514$ byte including the Ethernet header.

Packet timestamps at the probe sender and receiver are generated at points A and B, respectively, using libpcap at the hosts. We also use a specific endace DAG measurement card to obtain accurate reference timestamps. The timestamps are used to compute r_{in} and r_{out} for each packet train.

We generate two training data sets for a single tight link with exponential cross traffic. In data set (i) the capacity of the tight link is $C = 100$ Mbps. Exponential cross traffic with an average rate of $\lambda = 25$, 50, and 75 Mbps is used to generate different available bandwidths. In data set (ii) the capacity of the tight link is set to $C = 50$ Mbps and the exponential cross traffic has an average rate of $\lambda = 12.5$, 25, and 37.5 Mbps, respectively. In both cases the capacity of the access links is $C = 1$ Gbps. The probe streams comprise packet trains of $n = 100$ packets sent at $k = 20$ different rates with rate increment δ_r, where δ_r is determined as described in Section 3.1. For each configuration 100 repeated experiments are performed.

For training of the neural network, we first implement an autoencoder for each layer separately and then fine-tune the network using scaled conjugate gradient (scg). Given a regression network, we optimize the L2-error requiring approximately 1000 epochs until convergence is achieved. Training of the network (using Matlab) takes approximately 30 s. Due to the limited amount of training data (600 experiments overall in both training data sets), the shallow network with a small amount of hidden neurons allows training without much overfitting.

3.3. Evaluation: Exponential cross traffic, single tight link

We train the neural network using the two training data sets and generate additional data sets for testing. The test data is generated for the same network configuration as the training data set (i), i.e., using exponential cross traffic of 25, 50, and 75 Mbps at a single tight link of 100 Mbps capacity. We also consider other cross traffic rates of 12.5, 37.5, 62.5, and 87.5 Mbps that have not been included in the training data set (i) to see how well the neural network interpolates and extrapolates. We repeat each experiment 100 times so that we obtain 100 bandwidth estimates for each configuration. We compare the performance of the neural network-based method with the two model-based reference implementations of an iterative and a direct estimation method. All three methods generate available bandwidth estimates from the same measurement data.

3.3.1. Testing

The testing results of the neural network-based method are summarized in Fig. 4(a) compared to the results of the direct and the iterative method. We show the average of the available bandwidth estimates with error bars that depict the standard deviation of the estimates. The variability of the available bandwidth estimates is due to a number of reasons as discussed in Section 1.1. Particularly, the exponential cross traffic deviates from the fluid model and causes random fluctuations of the measurements of r_{out}.

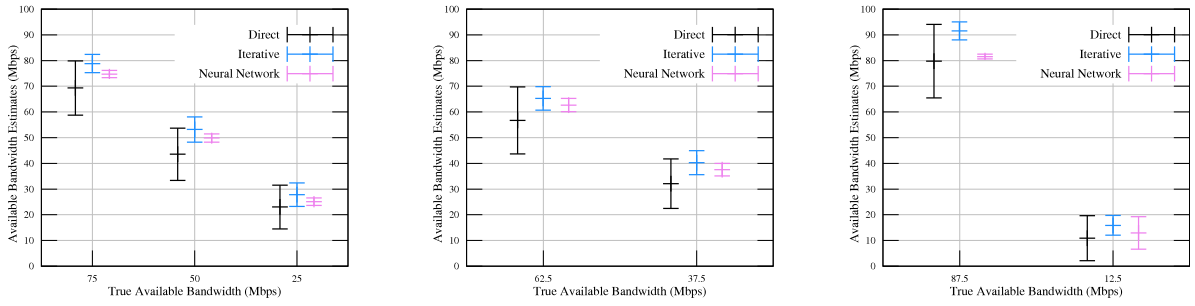
The variability of the available bandwidth estimates of the direct method is comparably large and the average underestimates the true available bandwidth. The iterative method shows less variability but tends to overestimate the available bandwidth. This is a consequence of the threshold test, where a lower threshold increases the responsiveness of the test but makes it more sensitive to random fluctuations. The neural network-based method improves the bandwidth estimates significantly. The average matches the true available bandwidth and the variability is low. The good performance of the neural network is not unexpected as it has been trained for the same network parameters.

3.3.2. Interpolation

Next, we consider cross traffic of the same type, i.e., exponential, however, with a different rate that has not been included in the training data. First, we consider cross traffic rates of 37.5 and 62.5 Mbps that fall into the range of rates 25, 50, and 75 Mbps that have been used for training, hence the neural network has to interpolate. The results in Fig. 4(b) show that the available bandwidth estimates of the neural network-based method are consistent in this case too.

3.3.3. Extrapolation

Fig. 4(c) depicts available bandwidth estimates for cross traffic rates of 12.5 and 87.5 Mbps. These rates fall outside the range of rates that have been included in the training data set so that the neural network has to extrapolate. The results of the neural network-based method are nevertheless highly accurate, with a noticeable underestimation of 5 Mbps on average only in case of a true available bandwidth of 87.5 Mbps. A reason for the lower accuracy that is observed when the available bandwidth approaches the capacity is that fewer measurements are on the characteristic upward line segment, see Fig. 2 that is also used for estimation by the direct method. The higher intensity of the cross traffic increases the variation in the available bandwidth estimates as can be seen in case of a true available bandwidth of 12.5 Mbps.



(a) Testing (b) Interpolation (c) Extrapolation

Fig. 4. Bandwidth estimates for different cross traffic rates that have been included in the training data set (testing), that fall into the range of the training data set (interpolation), and that fall outside the range of the training data set (extrapolation). The neural network-based method provides available bandwidth estimates that exhibit little variation and have an average that matches the true available bandwidth.

3.4. Network parameter variation beyond the training data

We investigate the sensitivity of the neural network with respect to a variation of network parameters that differ substantially from the training data set. Specifically, we consider two cases that are known to be hard in bandwidth estimation. These are cross traffic with high burstiness, and networks with multiple tight links. In this section, we evaluate the variability of the cross traffic. Multiple tight links are discussed in the following section.

3.4.1. Burstiness of cross traffic

To evaluate how the neural network-based method performs in the presence of cross traffic with an unknown burstiness, we consider three different types of cross traffic: constant bit rate (CBR) that has no burstiness as assumed by the probe rate model, moderate burstiness due to exponential packet inter-arrival times, and heavy burstiness due to Pareto inter-arrival times with infinite variance, caused by a shape parameter of $\alpha = 1.5$. The average rate of the cross traffic is $\lambda = 50$ Mbps in all cases. As before, the tight link capacity is $C = 100$ Mbps and the access links capacity is $C = 1$ Gbps.

The burstiness of the cross traffic can cause queueing at the tight link even if the probe rate is below the average available bandwidth, i.e., if $r_{in} < C - \lambda$. This effect is not captured by the fluid model. It causes a deviation from the ideal rate response curve as depicted in Fig. 2 that is maximal at $C - \lambda$ and blurs the bend that marks the available bandwidth. The result is an increase of the variability of available bandwidth estimates as well as an underestimation bias in both direct and iterative bandwidth estimation techniques [1,14].

Fig. 5 shows the mean and the standard deviation of 100 repeated experiments using the direct and iterative probing techniques and the neural network-based method. The average of the estimates shows a slight underestimation bias compared to the true available bandwidth if the cross traffic burstiness is increased. More pronounced is the effect of the cross traffic burstiness on the standard deviation of the bandwidth estimates. While for CBR cross traffic the estimates are close to deterministic, the variability of the estimates increases significantly if the cross traffic is bursty. The neural network, that has been trained for exponential cross traffic only, performs almost perfectly in case of CBR cross traffic and shows good results with less variability compared to the direct and iterative techniques also for the case of Pareto cross traffic.

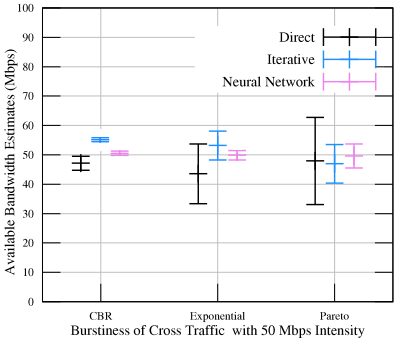


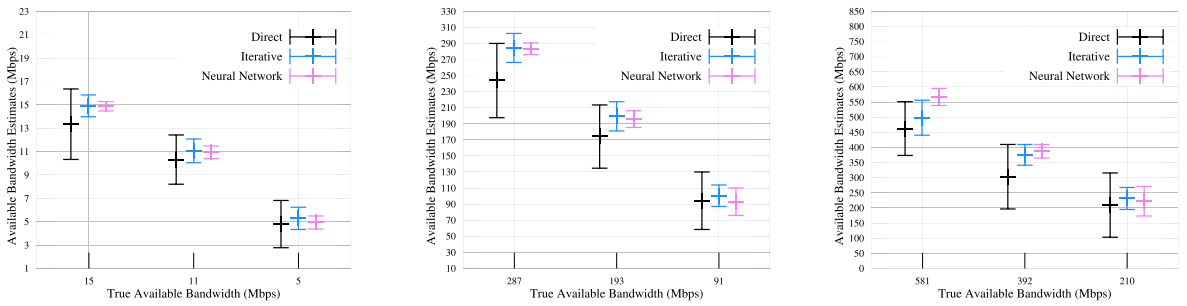
Fig. 5. Bandwidth estimates for different types of cross traffic burstiness. An increase of the burstiness causes a higher variability of the bandwidth estimates as well as an underestimation bias.

4. Variation of the tight link capacity and multiple tight links

In this section, we evaluate the performance of our neural network-based method for a wide range of different network scenarios. For example, the capacity of the network may be small, e.g., in access networks, or large, e.g., in backbone networks. Further, these networks may have a high cross traffic intensity with an unknown burstiness and possibly multiple tight links. Hence, the test data may differ substantially from the training data. For the purpose of training, we use data sets (i) and (ii) generated for a single-hop network with tight link capacity $C = 50$ Mbps and $C = 100$ Mbps, respectively, in the presence of exponential cross traffic, i.e., moderate burstiness. For testing we use data sets that have much more variation than the training data. The testing data is obtained using randomly generated networks that cover a wide range of capacities and cross traffic intensities with unknown burstiness. We also address the known underestimation bias in case of multiple tight links and include networks where the tight link does not coincide with the bottleneck link in our evaluation.

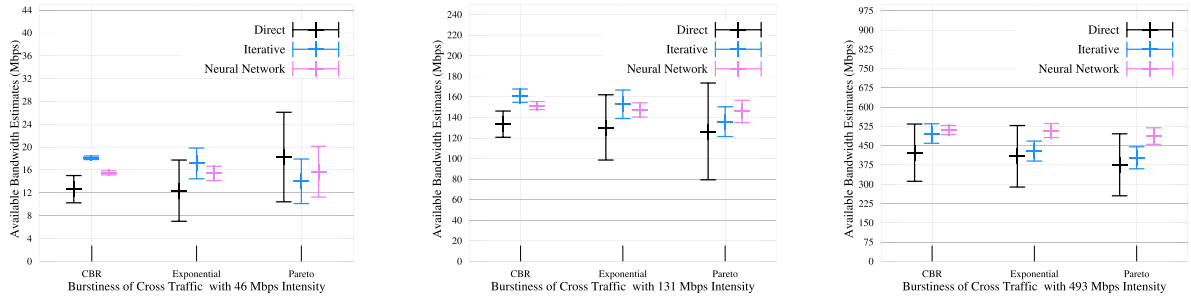
4.1. Random networks

Since our implementation of the neural network-based method is scale-invariant (within the limits of the fluid model), we expect that the method can perform bandwidth estimation in arbitrary networks. To investigate the sensitivity of our method with respect to different network parameters, we set up a number of topologies where the tight link capacity and the cross traffic intensity are chosen randomly. The capacity is a random number in the interval [10 Mbps, 1 Gbps] with exponential distribution, i.e., $C = 10^{U[1,3]}$ Mbps where $U[1,3]$ is a uniform random variable in the interval $[1,3]$. The cross traffic intensity is chosen relative to the tight link capacity as $\lambda = U[0,1] \cdot C$ Mbps. The link and traffic characteristics are emulated as described in Section 3.2. The access link capacity for all networks is $C = 1$ Gbps. Since the capacity is unknown to the estimation method, the probing increment δ_r is chosen after sending initial packet train probes as a burst as described in Section 3.1.



(a) Bandwidth estimates in C = 22 Mbps (b) Bandwidth estimates in C = 385 Mbps (c) Bandwidth estimates in C = 831 Mbps

Fig. 6. Available bandwidth estimates for a wide range of intensities of exponential cross traffic in random networks with a tight link capacity of (a) C = 225 Mbps, (b) C = 383 Mbps and (c) C = 831 Mbps.



(a) Bandwidth estimates in C = 61 Mbps (b) Bandwidth estimates in C = 276 Mbps (c) Bandwidth estimates in C = 1 Gbps

Fig. 7. Available bandwidth estimates for unknown cross traffic burstiness in random networks with a tight link capacity C and true available bandwidth A: (a) C = 61 Mbps, A = 15 Mbps (b) C = 276 Mbps, A = 145 Mbps and C = 1 Gbps, A = 507 Mbps.

4.1.1. Available bandwidth estimates

We perform a number of experiments using randomly generated networks with a wide variety in the tight link capacity, cross traffic intensity, and cross traffic burstiness. We consider small, moderate and large values of these network parameters to evaluate the performance of our proposed method. First, we consider different intensities of exponential cross traffic and second, different cross traffic burstiness: CBR (least or without), exponential (moderate) and Pareto (heavy) in random networks.

We show some selected results which are representative of the performance of our approach. Figs. 6(a)–6(c) show the results for random networks with a tight link capacity of C = 22 Mbps (small), C = 385 Mbps (medium), and C = 831 Mbps (large). Further, corresponding to each tight link capacity, the cross traffic intensities also vary from low to high. For an example, Fig. 6(a) shows the results for the data obtained for a network with a tight link capacity C = 22 Mbps in the presence of exponential cross traffic with intensity λ = 7 Mbps (low), λ = 11 Mbps (medium), and λ = 17 Mbps (high). Similarly, for Figs. 6(b) and 6(c) the data is obtained for networks with capacity C = 385 Mbps and C = 831 Mbps, and the corresponding exponential cross traffic of intensities 98, 192, 294 Mbps and 250, 439, 621 Mbps, respectively.

Figs. 7(a)–7(c) show results in the presence of cross traffic with different types of burstiness for different randomly generated networks. The data is obtained for networks with a tight link capacity of C = 61 Mbps (small), C = 276 Mbps (medium), and C = 1 Gbps (large) with an average rate of the cross traffic of 46, 131, and 493 Mbps.

The results confirm that our method is able to estimate the available bandwidth in networks with arbitrary capacities and in the presence of different intensities and unknown burstiness of the cross traffic. The average of the available bandwidth estimates corresponds to the true available bandwidth, though the variation increases with higher burstiness and intensity of the cross traffic.

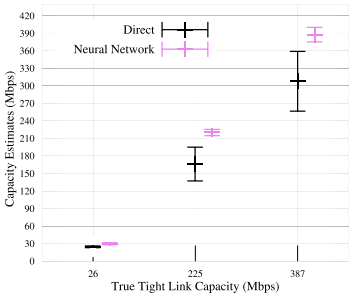


Fig. 8. Tight link capacity estimates for random networks in the presence of exponential cross traffic with an average intensity of 4, 60, and 98 Mbps.

4.1.2. Capacity estimates

The output of our neural network-based approach is the tuple of bottleneck capacity and available bandwidth estimates. We evaluate our method to estimate the tight link capacity of networks. The measurement data is obtained for random networks with a true tight link capacity of C = 26 Mbps, C = 225 Mbps, and C = 387 Mbps in the presence of exponential cross traffic of intensity 14, 60, and 98 Mbps, respectively.

Fig. 8 depicts the capacity estimates obtained by the direct method and the neural network, respectively. Both methods perform well for small capacities. However, in higher capacity networks, the variation of the estimates increases. This is due to the fact that the inter packet gaps are highly susceptible to noise introduced by bursty cross traffic in these networks. The difficulty to attain precise smaller input gaps g_{in} makes the input data noisy. As described in Section 2.2, the direct method estimates the capacity from the slope of the upward segment of a rate response curve shown in Fig. 2. In the presence of random cross traffic the slope gets altered. This leads to underestimation and variability in the estimates of the direct method. However, the neural

network, that is trained only for small capacities of 50 Mbps and 100 Mbps, is able to perform well even in higher capacity networks. The estimates are accurate with a mean value matching the true capacity value, though the variation of the estimates increases with increasing capacity due to noisy test data. Thus, with our proposed method we are able to accurately estimate the available bandwidth as well as the capacity.

4.2. Multiple tight links

We now address networks with multiple tight links. These networks pose a well known challenge in available bandwidth estimation. In case of multiple tight links, the probe stream has a constant rate r_{in} with a defined input gap g_{in} only at the first link. For the following links, the input gaps have a random structure as they are the output gaps of the preceding links. At each additional link the probe stream interacts with new, bursty cross traffic. This causes lower probe output rates and results in underestimation of the available bandwidth in multi-hop networks [1,13–15].

To test the neural network with multiple tight links, we extend our network from single-hop to multi-hop as shown in Fig. 3. The path-persistent probe streams experience single hop-persistent exponential cross traffic with average rate $\lambda = 50$ Mbps while traversing multiple tight links of capacity $C = 100$ Mbps. The capacity of the access links is 1 Gbps.

In Fig. 9(a) we show the results from 100 repeated measurements for networks with 1 up to 4 tight links. The model-based methods, both direct and iterative, as well as the neural network-based method underestimate the available bandwidth with increasing number of tight links. The reason for underestimation in case of model based techniques is the cross traffic burstiness which potentially reduces the output probe rate at each of the tight links. Though the estimates of our neural network show the least variability, the neural network underestimates the available bandwidth. This is not surprising, as it is trained only for a single tight link.

4.2.1. Training for multiple tight links

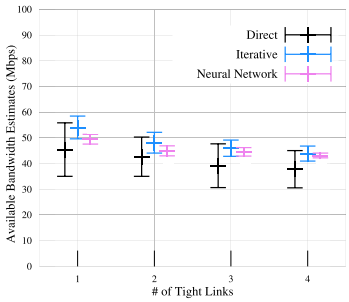
Since the neural network, that is trained only for a single tight link, underestimates the available bandwidth in case of multiple tight links, we consider training the neural network for the latter. A neural network has also been trained for multiple tight links in [23], using, however, packet chirp probes. We extend the single tight link network up to four tight links as shown in Fig. 3.

To generate the training data, a probe stream consisting of k packet trains is sent through a network with two, three, and four tight links, respectively, each with exponential cross traffic with an average rate $\lambda = 50$ Mbps. In all networks the probe stream is path persistent and the cross traffic is single hop-persistent. The capacity of the access links is 1 Gbps and that of the tight links is 100 Mbps. The neural network is trained with this additional training set (iii) for multiple tight links along with training sets (i) and (ii) for a single tight link.

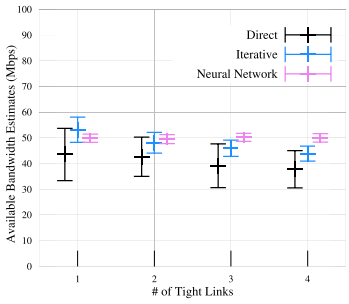
Fig. 9(b) compares the results of direct and iterative methods with the neural network. As can be seen clearly, the results have improved significantly with the neural network after it has been trained for multiple tight links. The mean value matches the true available bandwidth and the estimates have less variability.

4.2.2. Tight link differs from bottleneck link

The problem of available bandwidth estimation in multi-hop networks becomes more difficult if the tight link of a network path differs from the bottleneck link. As shown in Fig. 10(a), link i is the *bottleneck link*, i.e., the link with the minimal capacity, and link $i+1$ represents the *tight link*, i.e., the link that has the minimal available bandwidth [2]. The existence of separate tight and bottleneck links has an impact on the shape of the rate response curve. The rate response curve in Fig. 2 is derived under the assumption that there is a single tight link that



(a) Bandwidth estimates with the neural network trained for a single tight link



(b) Bandwidth estimates with the neural network trained for multiple tight links

Fig. 9. Multiple tight links with capacity $C = 100$ Mbps in the presence of single hop-persistent exponential cross traffic with an average rate $\lambda = 50$ Mbps. (a) All methods tend to underestimate the available bandwidth in case of multiple tight links. (b) The training of the neural network for multiple tight links improves the bandwidth estimates significantly.

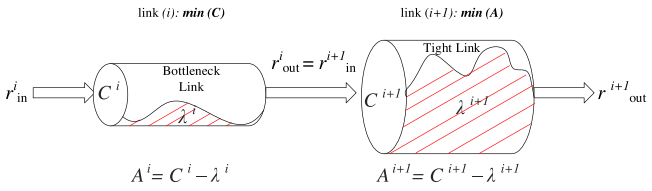
matches the bottleneck link. If this assumption does not hold, available bandwidth estimation by direct methods becomes more difficult. We investigate the impact further by considering two different scenarios. In the first scenario, i.e., scenario I, the bottleneck link appears in front of the tight link. In the second, i.e., scenario II, it is vice versa. In both the scenarios as seen in Fig. 10(a) only the input rate r_{in}^i at the first link i has the desired input gap. As the packet train traverses through the first link with capacity C^i, it interacts with the cross traffic with average rate λ^i and the resulting output rate r_{out}^i is obtained by rearranging Eq. (4) as

$$r_{\text{out}}^i = \begin{cases} r_{\text{in}}^i & \text{if } r_{\text{in}}^i \le C^i - \lambda^i, \\ \frac{r_{\text{in}}^i C^i}{r_{\text{in}}^i + \lambda^i} & \text{if } r_{\text{in}}^i > C^i - \lambda^i. \end{cases} \tag{6}$$

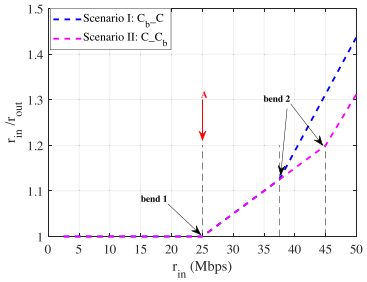
For the next link $i+1$, the input has a different rate as it is fed from the first link i, i.e., $r_{\text{in}}^{i+1} = r_{\text{out}}^i$. The output rate r_{out}^{i+1} from hop $i+1$ is obtained by recursive insertion of Eq. (6) as

$$r_{\text{out}}^{i+1} = \begin{cases} r_{\text{in}}^i & \text{if } r_{\text{in}}^i \le A^{i+1} \;,\; r_{\text{in}}^i \le A^i, \\ \frac{r_{\text{in}}^i C^{i+1}}{r_{\text{in}}^i + \lambda^{i+1}} & \text{if } r_{\text{in}}^i > A^{i+1} \;,\; r_{\text{in}}^i \le A^i, \\ \frac{r_{\text{in}}^i C^i}{r_{\text{in}}^i + \lambda^i} & \text{if } \frac{r_{\text{in}}^i C^i}{r_{\text{in}}^i + \lambda^i} \le A^{i+1} \;,\; r_{\text{in}}^i > A^i, \\ \frac{\frac{r_{\text{in}}^i C^i}{r_{\text{in}}^i + \lambda^i} C^{i+1}}{\frac{r_{\text{in}}^i C^i}{r_{\text{in}}^i + \lambda^i} + \lambda^{i+1}} & \text{if } \frac{r_{\text{in}}^i C^i}{r_{\text{in}}^i + \lambda^i} > A^{i+1} \;,\; r_{\text{in}}^i > A^i. \end{cases} \tag{7}$$

where C^i and C^{i+1} are the link capacities, λ^i and λ^{i+1} are the single hop-persistent cross traffic intensities, and $A^i = C_i - \lambda_i$ and $A^{i+1} =$



(a) Tight link differs from bottleneck link



(b) Rate response curves for a two-hop network

Fig. 10. (a) A two-hop network where the tight link differs from the bottleneck link. (b) Rate response curves of a two-hop network representing scenario I and II where the tight link occurs behind and in front of the bottleneck link, respectively. The presence of two bends indicates that two links are congested.

$C_{i+1} - \lambda_{i+1}$ are the corresponding available bandwidths for link i and $i+1$, respectively. Eq. (7) represents the two scenarios defined above. In the case of scenario I, where link $i+1$ is the tight link, lines 1, 2, and 4 of Eq. (7) apply, i.e., the probe stream r^i_{in} is shaped at none, the second, or both links. Conversely, in scenario II, where link i is the tight link, lines 1, 3, and 4 apply.

Fig. 10(b) shows the rate response curves for the two scenarios obtained using Eq. (7). The curves are piece-wise linear. The two bends indicate the presence of two congestible links. For both scenarios, the tight link capacity is $C = 100$ Mbps and the bottleneck capacity is $C_b = 50$ Mbps. The cross traffic is CBR with an average rate of $\lambda = 75$ Mbps and $\lambda_b = 12.5$ Mbps, traversing the tight link and the bottleneck link respectively.

Considering Fig. 10(b), we have $r^{i+1}_{out} = r^i_{in}$ until the probe rate reaches the first bend at 25 Mbps where the available bandwidth of the tight link is saturated. In scenario I, when we increase the probe rate further, we reach a second bend at 37.5 Mbps where the available bandwidth of the bottleneck link, i.e., link i, is also saturated. However, if the order of the two-hop network is reversed, i.e., the bottleneck link succeeds the tight link, the rate response curve differs as represented by scenario II. The reason is that the output of the tight link, i.e., link i in this scenario, is shaped so that r^i_{out} exceeds 37.5 Mbps, that is the available bandwidth of the bottleneck link, i.e., link $i+1$, only if r^i_{in} exceeds 45 Mbps.

In both scenarios, the second linear segment of the rate response curve can cause overestimation of the available bandwidth if a direct probing method is used. This issue is addressed in [3] by performing a linear regression to each of the linear segments under the assumption that the tight link precedes the bottleneck link in the network. In practical scenarios, however, the order of tight and bottleneck links is a priori unknown. Iterative probing techniques, on the other hand, can still estimate the available bandwidth, even if the tight link and the bottleneck link differ, by searching for the first turning point of the rate response curve, i.e., by sending repeated probes at increasing rates, as long as $r^{i+1}_{out} = r^i_{in}$.

To test the performance of our neural network-based method we extend our network to a two-hop network, where the tight link and the bottleneck link do not coincide. The access link capacity is $C = 1$ Gbps. We generate test data for the two scenarios described above. Further, we also consider bursty cross traffic along with CBR.

Figs. 11(a) and 11(c) show the available bandwidth and Figs. 11(b) and 11(d) the bottleneck capacity estimates in scenario I and II, respectively. The neural network is trained for a single tight link only, using the training data set (i) and (ii). The results obtained from the neural network are more consistent as compared to direct and iterative probing methods. However, a bias can be noticed in the available bandwidth and the bottleneck capacity estimates, specifically for scenario II in Figs. 11(c) and 11(d), i.e., where the tight link precedes the bottleneck link. This can be explained by the deviation of the output rate r^{i+1}_{out} as obtained from Eq. (7) compared to Eq. (6) which affects the input feature vector of r_{in}/r_{out} of the neural network.

To reduce the bias in the estimation, we consider training the neural network for networks where the tight link is different from the bottleneck link. An additional training set (iv) is generated in a two-hop network for both scenarios: I and II, where the tight link follows the bottleneck link and precedes the latter, respectively.

Fig. 12 shows the available bandwidth and the bottleneck capacity estimates that are obtained after additional training for scenarios I and II. As can be seen in Fig. 12(d), the bias of the capacity estimates is reduced significantly.

5. Iterative neural network-based method

State-of-the-art iterative probing methods perform a search for the available bandwidth by varying the probe rate r_{in} until r_{in} converges to the available bandwidth. Pathload [6] uses statistical tests to determine whether r_{in} exceeds the available bandwidth or not and performs a binary search to adapt r_{in} iteratively. The recent method [17] adopts Pathload's binary search algorithm but uses machine learning instead of statistical tests to determine whether r_{in} exceeds the available bandwidth or not. Our proposed iterative neural network-based method differs from [17] in several respects.

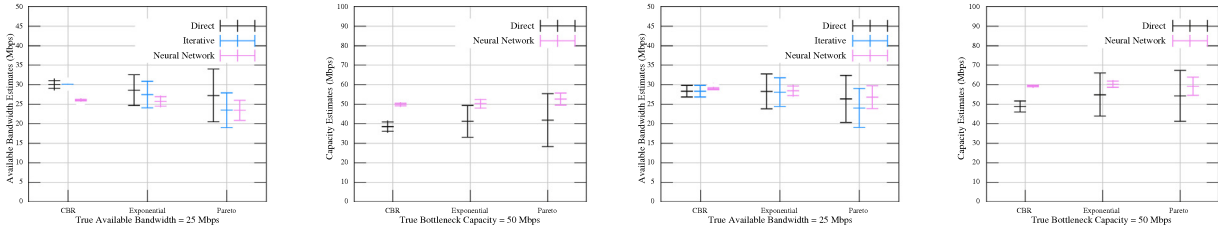
We propose an iterative neural network-based method that (a) determines the next probe rate by a neural network, that is trained to select the probe rate that improves the bandwidth estimate most, instead of using the binary search algorithm, and (b) it includes the information of all previous probe rates to estimate the available bandwidth instead of considering only the current probe rate. Our implementation comprises two parts. First, we train the neural network to cope up with input vectors that are not fully populated. Second, we create another neural network that recommends the most beneficial probe rates.

5.1. Partly populated input vectors

An iterative method will only use a limited set of probe rates. Correspondingly, we mark the entries of the input vector that have not been measured as invalid by setting $r_{in}/r_{out} = 0$. To obtain a neural network that can deal with such partly populated input vectors, we perform training using the training data sets (i) and (ii), where we repeatedly erase a random number of entries at random positions. While testing the neural network we erase entries in the same way.

In Fig. 13, we show the absolute error of the available bandwidth estimates that are obtained by the neural network if $m \in [1, k]$ randomly selected entries of the k-dimensional input vector are given. The bars show the average error and the standard deviation of the error. The data set used for testing is the same as the one used for Fig. 4(a) previously, i.e., $C = 100$ Mbps and $A \in [25, 50, 75]$ Mbps. We show the combined results for all values of A.

The average error shows a clear improvement with increasing m. For $m = 1$, the information is not sufficient to identify the two unknown parameters capacity and available bandwidth. Hence, the neural network first reports conservative estimates in the middle range. For comparison, by guessing 50 Mbps in all cases the average error is 16.6 Mbps for the given test data set. With increasing m the neural



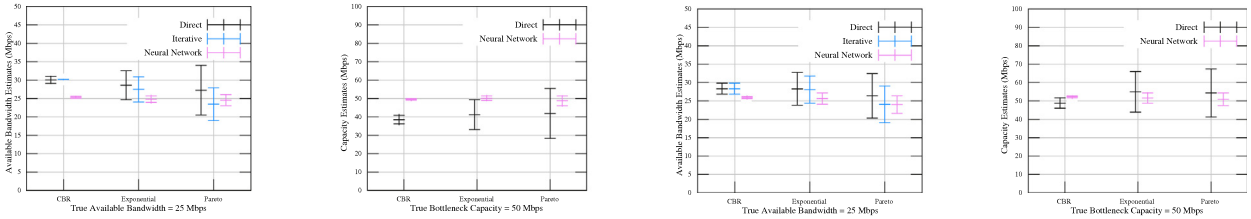
(a) Bandwidth estimates in scenario I trained for a single tight link

(b) Capacity estimates in scenario I trained for a single tight link

(c) Bandwidth estimates in scenario II trained for a single tight link

(d) Capacity estimates in scenario II trained for a single tight link

Fig. 11. Estimates from the neural network trained for a single tight link: (a) available bandwidth and (b) capacity estimates of a two-hop network, where the tight link follows the bottleneck link and, (c) available bandwidth and (d) capacity estimates where the tight link precedes the latter. In both the scenarios, the tight link capacity is $C = 100$ Mbps and the bottleneck capacity is $C_b = 50$ Mbps with cross traffic intensity of $\lambda = 75$ Mbps and $\lambda_b = 12.5$ Mbps respectively.



(a) Bandwidth estimates in scenario I trained for the tight link different from the bottleneck link

(b) Capacity estimates in scenario I trained for the tight link different from the bottleneck link

(c) Bandwidth estimates in scenario II trained for the tight link different from the bottleneck link

(d) Capacity estimates in scenario II trained for the tight link different from the bottleneck link

Fig. 12. Estimates from the neural network trained for different tight link and bottleneck link: (a) available bandwidth and (b) capacity estimates of a two-hop network, where the tight link follows the bottleneck link and, (c) available bandwidth and (d) capacity estimates where the tight link precedes the latter. In both the scenarios, the tight link capacity is $C = 100$ Mbps and the bottleneck capacity is $C_b = 50$ Mbps with cross traffic intensity of $\lambda = 75$ Mbps and $\lambda_b = 12.5$ Mbps respectively.

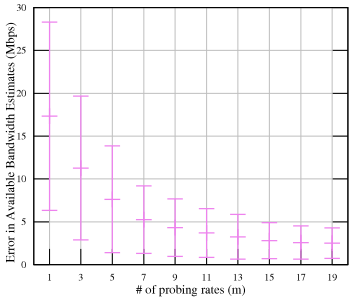


Fig. 13. Error of the available bandwidth estimates obtained for a set of m randomly selected probe rates.

network starts to distinguish the range of $A \in [25, 50, 75]$ Mbps but tends to frequent misclassifications that can cause large errors. These misclassifications are mostly resolved when increasing m further.

We observe the same trend also for the error of the capacity estimates that shows a high correlation with the error of the available bandwidth estimates. Hence, we omit showing the results.

5.2. Recommender network for probe rate selection

When adding entries to the partly populated input vector of the neural network, the average estimation error improves. The amount of the improvement depends, however, on the position of the a priori unknown entry that is added, as well as on the m entries that are already given, i.e., their position and value. We use a second neural network that learns this interrelation. Using this knowledge, the neural network acts as a recommender that given a partly populated input vector selects the next probe rate, i.e., the next entry, that is expected to improve the accuracy of the bandwidth estimate the most. The recommender network takes the $k = 20$-dimensional input vector of values $r_{\text{in}}/r_{\text{out}}$, has 40 hidden neurons, and generates a k-dimensional output vector of estimation errors that apply if the entry $r_{\text{in}}/r_{\text{out}}$ is added at the respective position. Given the output vector, the rate r_{in} that minimizes the estimation error is selected for probing next.

Fig. 14 shows how the recommender network improves the error of the bandwidth estimates compared to the random selection of probe rates in Fig. 13. Starting at 5 selected probe rates, the average estimation error as well as the standard deviation of the error are small and adding further probe rates improves the estimate only marginally. The reason is that certain probe rates, e.g., those on the horizontal line at $r_{\text{in}}/r_{\text{out}} = 1$ in Fig. 2, provide little additional information. We conclude that the recommender can effectively control the selection of probe rates to avoid those rates that contribute little. In this way, the recommender can save a considerable amount of probe traffic.

5.3. Neural network as available bandwidth classifier

Our iterative neural network-based approach as proposed in Section 5.2 differs from state-of-the-art machine learning techniques for bandwidth estimation in several respects. For a comparison, we formulate available bandwidth estimation as a classification problem. Specifically, to compare our approach with state-of-the-art technique [17], we implement two different classifiers: an individual classifier which

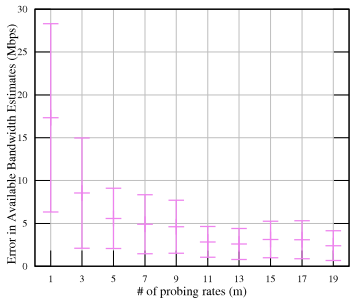


Fig. 14. Error of the available bandwidth estimates obtained for a set of m recommended probe rates.

represents state-of-the-art and uses information only from the current probe rate to determine whether the current probe rate is above or below the available bandwidth, and a full classifier which is based on our proposed method and uses a k-dimensional feature vector with information of all previous probe rates to classify the probe rates. We test both classifiers with the measurement data generated in the presence of bursty cross traffic with an average rate of 50 Mbps. The tight link and the access link capacity is 100 Mbps and 1 Gbps as before.

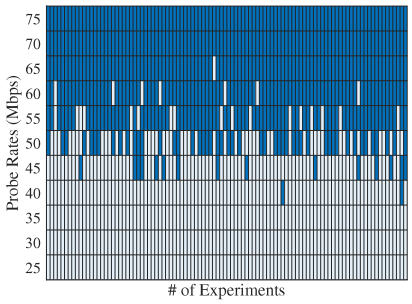
5.3.1. Individual classifier

The individual classifier represents state-of-the-art and uses information of a single probe rate to classify whether the current rate is above or below the available bandwidth. The goal is to increment the probe rates iteratively by δ_r until the available bandwidth is saturated. For our reference implementation, we decided for a shallow neural network consisting of one hidden layer with 40 neurons. The input feature vector is the ratio of current probe rate at the sender and at the receiver, i.e., $r_{\text{in}}/r_{\text{out}}$. The neural network has one output neuron that represents the two different categories: current probe rate *below* or *above* the available bandwidth. Depending upon the result, the next probe rate is increased by δ_r iteratively until the probe rate r_{in} exceeds the available bandwidth.

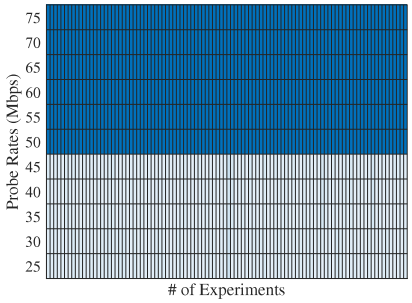
Fig. 15(a) shows the results of the individual classifier for 100 experiments in the form of a classification map in the probe range $[25, \ldots, 75]$ Mbps. The true available bandwidth is 50 Mbps. The probe rates which are classified as being above the available bandwidth are shown by blue pixels, and those which are classified as below the available bandwidth are represented by white pixels. However, due to the burstiness of the cross traffic, there exist false positive errors, i.e., certain rates below the available bandwidth are misclassified as above. These misclassifications can be seen by occurrence of blue pixels below 50 Mbps. The precision value is 0.97. Similarly, there are false negatives, i.e., certain rates above the available bandwidth are misclassified as below. This can be seen by white pixels re-occurring above the true available bandwidth of 50 Mbps. This leads to an inconsistency in the classification with the recall value of 0.87. It is not clear from the classification map which exact probing rate saturates the available bandwidth. This is why Pathload does not report a single available bandwidth estimate but an available bandwidth region [6].

5.3.2. Full classifier

To compare with state-of-the-art and motivated by the limitations of the individual classifier, we use our proposed method to implement a full classifier. The full classifier uses a k-dimensional vector of values $r_{\text{in}}/r_{\text{out}}$ as input, i.e., it does not operate iteratively on individual values of $r_{\text{in}}/r_{\text{out}}$ but uses all k values of $r_{\text{in}}/r_{\text{out}}$ at once. The classifier is implemented by a shallow neural network consisting of one hidden layer with 40 neurons. The output of the neural network is k-dimensional vector consisting of binary values that classify whether the respective probe rate is above or below the available bandwidth.



(a) Individual Classifier



(b) Full Classifier

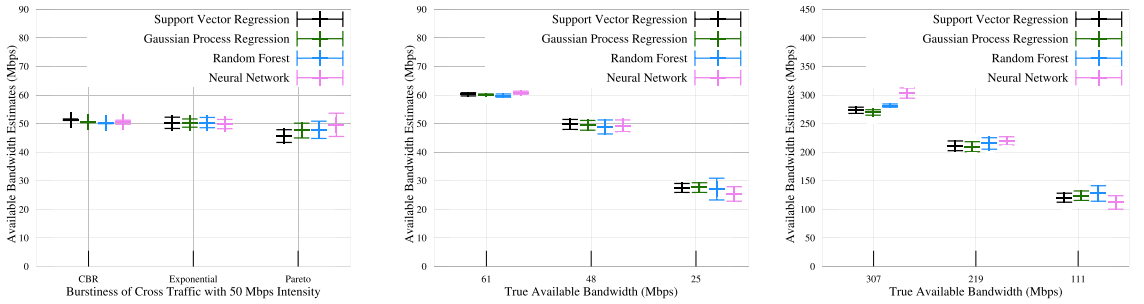
Fig. 15. The classification map shows the results of (a) the individual, and (b) the full classifier. The tight link capacity is $C = 100$ Mbps. In the presence of exponential cross traffic with average rate $\lambda = 50$ Mbps, the true available bandwidth is 50 Mbps. The blue pixels indicate the probe rates that are classified as above the available bandwidth and the white pixels indicate the vice versa.

Fig. 15(b) shows the classification map results for the full classifier. The results are highly accurate without any misclassification with precision and recall values of 1.0, respectively. This can be seen by occurrence of only blue pixels at probe rates which exceed the true available bandwidth of 50 Mbps. The results confirm the stability of our proposed method in comparison to the individual classifier. The consistency of results comes from the fact that instead of using only the current probe rate, the information of all the previous probe rates is included. The ability to classify the probe rates directly, instead of doing the binary search as done in [17], increases the computational speed of estimation.

5.4. Evaluation of other machine learning techniques

Regarding the parameters of our neural network, we also explored the use of deeper networks with more hidden layers, convlayers and residual networks. However, in our setting different variants of networks did not improve the quality in our experiments. We believe, that the main reason is overfitting which is caused by the sparse amount of data used for training.

To evaluate our proposed method with other supervised machine learning techniques, we consider three machine learning algorithms that can do a linear regression to estimate the available bandwidth. Two of the chosen techniques are kernel based (1) Support Vector Regression (2) Gaussian Process Regression, whereas the third one is ensemble-based (3) Random Forest. We provide an overview of these techniques and justify their selection for available bandwidth estimation in the section below. Our data set consists of 600 observations: training set (i) and (ii). Due to sparse amount of training data, we use 5-fold cross validation to evaluate our machine learning algorithms, i.e., how accurately they are able to estimate the available bandwidth for unseen test data. We used parallel computing in Matlab R2017b on a quad-core Linux-machine, with processor base frequency of 3.20 GHz running Ubuntu 14.04 LTS and kernel of version 4.4.0-130-generic.



(a) Bandwidth estimates in C = 100 Mbps (b) Bandwidth estimates in C = 89 Mbps (c) Bandwidth estimates in C = 417 Mbps

Fig. 16. Available bandwidth estimates (a) in the presence of bursty cross traffic with the average rate $\lambda = 50$ Mbps and $C = 100$ Mbps, (b) in the random network with the moderate capacity $C = 89$ Mbps , and (c) higher capacity $C = 417$ Mbps in the presence of wide range of exponential cross traffic. The results for neural networks reduce bias and variance even at higher capacities.

5.4.1. Support vector regression

Support Vector Regression (SVR) is a non-parametric kernel based machine learning technique that performs linear regression for non-linear functions in the high-dimension feature space using ϵ-insensitive loss. We use the "Gaussian (Radial Basis Function)" kernel and epsilon is set $\epsilon = 0.55$, using the interquartile range of the response variable. In ϵ-SV regression, the goal is to find a function $f(x)$ that has at most a deviation of ϵ from the actually obtained targets for all the training data. The ϵ-insensitive loss function optimizes the generalization bounds given for regression, ignoring the errors as long as they are situated within a certain distance of the true value. We selected SVM as it has a regularization parameter that avoids over-fitting of data and has excellent generalization capability, with high prediction accuracy. The computational complexity of SVR does not depend on the dimensionality of the input space. The training time is approximately 30 s with the prediction speed of 7900 obs/s.

5.4.2. Gaussian Process regression

Gaussian Process Regression (GPR) is a non-parametric kernel-based probabilistic machine learning technique. We are interested in making inferences about the relationship between inputs and targets, i.e., the conditional distribution of the targets given the input. In Gaussian processes, the covariance function which expresses the similarity between the two inputs plays a crucial role [30]. GPR seeks to determine the optimal values of the hyper-parameters governing the covariance function. We use an "exponential" kernel to define the covariance matrix. A prior over functions is defined, which is converted into a posterior over functions once we have observation data. The observation data causes the resulting posterior GP samples to be constrained to pass near the observed data points. For an example, we have a k-dimensional training feature vector x for which we have observed the function f that estimates the available bandwidth. Now, given a new k-dimensional test vector x_*, we estimate the new function f_* which follows the probability distribution $p(f_*/x_*, x, f)$ where f and f_* have a joint Gaussian distribution. As compared to SVR, GPR provides a probabilistic prediction and an estimate of uncertainty in the prediction. With GPR using an exponential kernel function, the prediction speed is 11000 obs/s and training time is approximately 38 s.

5.4.3. Random forest

Random Forest (RF) is an ensemble machine learning approach used for regression, in which "weak learners" collaborate to form "strong learners" trained with the Bootstrap Aggregation (Bagging) method. Bootstrap Aggregation, as suggested by its name, uses a bootstrapping technique to sample the input data randomly with replacement to create multiple subsets of data. A set of models is then trained on each of these subsets, and their predictions are aggregated to make the final combined prediction using averaging. The averaging of multiple decision trees reduces the variance and bias and avoids over-fitting. The performance of the ensemble technique depends on the setting of the ensemble and of the weak learners. A higher number of trees in the random forest increases the performance and makes the predictions more stable, but it also slows down the computation, making it ineffective for real-time bandwidth predictions. We chose the hyper-parameters to have a trade-off between the predictive power and the computational speed and chose the number of ensemble learning cycles to be 30. It took approximately 31 s to train with the predicting speed of 2700 obs/s. To minimize the generalization error, the minimum number of leafs that are required to split an internal node is set to 8.

5.4.4. Evaluation

To evaluate our method, we use the measurement data obtained from the network in the presence of cross traffic of unknown burstiness with intensity $\lambda = 50$ Mbps and from random networks. The access link capacity for both the networks is $C = 1$ Gbps. The bottleneck capacity is $C = 100$ Mbps in the former network, whereas the capacity and cross traffic are chosen randomly in the latter. As can be seen in Figs. 16(a) and 16(b), our proposed method works using supervised machine learning techniques as well. These techniques are based on the assumption that if two training feature vectors are close, then their corresponding output prediction functions should be also close. With the local generalization principle, they are able to interpolate locally. However, this principle has its limitations. Fig. 16(c) shows the results where these techniques reduce the variance but a slight bias is visible in bandwidth estimation as compared to the neural network. This is due to the fact that the test data has higher variations as compared to the training data. The measurement data is obtained for a random network with a higher capacity $C = 417$ Mbps where it is difficult to achieve a precise inter packet gap g_{in}. These distortions in input gaps result in noisy data, i.e., higher variations which are not covered in the training data obtained for a network with bottleneck capacity $C = 50$ Mbps and $C = 100$ Mbps, respectively. However, with the neural networks the mean of 100 iterations matches the true available bandwidth and the estimates have low variability. The neural network can generalize non-locally which kernel or ensemble machines with standard generic kernels are not able to do. It has the ability to recognize complicated functions even in the presence of noise and variability. We are able to reduce the bias and variance in available bandwidth estimates even using a shallow neural network. For these reasons we have chosen the neural network for bandwidth estimation.

6. Conclusion

We investigated how neural networks can be used to benefit measurement-based available bandwidth estimation. We proposed a method that is motivated by the characteristic rate response curve of a network. Our method takes a vector of ratios of equidistantly spaced

probe rates at the sender and at the receiver r_{in}/r_{out} as input to a neural network to estimate the available bandwidth and the bottleneck capacity. We use ratios of data rates and a suitable normalization to achieve an implementation that is scale-invariant with respect to the network capacity. We conducted a comprehensive measurement study in a controlled network testbed. Our results showed that neural networks can significantly improve available bandwidth estimates by reducing bias and variability. This holds true also for network configurations that have not been included in the training data set, such as different types and intensities of cross traffic and randomly generated networks where the capacity and cross traffic intensity can vary substantially. We also included network scenarios with multiple tight links, and multi-hop networks where the tight link differs from the bottleneck link. While the estimates of the neural network were also good in these scenarios, additional training accomplished a noticeable improvement. To reduce the amount of probe traffic, we implemented an iterative method that varies the probe rate adaptively. The selection of probe rates is performed by a neural network that acts as a recommender. The recommender effectively selects the probe rates that reduce the estimation error most quickly. To compare with state-of-the-art, we formulated available bandwidth estimation as a classification problem. We compared our full classifier based on our proposed method, using information of all previous probe rates with state-of-the-art individual classifier based on only the current probe rate. We evaluated our method with other supervised machine learning techniques. The ability of the neural network to generalize non-locally allows it to reduce the bias and variance even in the presence of noisy feature vectors.

7. Data set

We provide benchmark data sets consisting of k-dimensional matrices of ratios of r_{in}/r_{out} which are input to the neural network, and r_{in} to calculate δ_r, with respect to which the available bandwidth and bottleneck capacity are normalized. The first row of data sets has true bottleneck capacity and available bandwidth values as output for training the neural network. The data sets are generated considering different network parameters, and address the problems known to be difficult in the bandwidth estimation. To evaluate the scale-invariance approach of our proposed method with respect to network capacity, we have generated two data sets with different single tight link capacity. The problem of underestimation of available bandwidth in a multi-hop network is addressed using the third data set for multiple tight links. Further to increase the difficulty of the problem in a multi-hop network, the fourth data set is generated for networks where the tight link is different from the bottleneck link. The data sets can be downloaded from [31]. The first data set consists of samples for a single tight link with the capacity $C = 100$ Mbps and exponential cross traffic with an average rate of $\lambda = 25$, 50, and 75 Mbps. The second data set is also for a single tight link but with the different capacity $C = 50$ Mbps and the exponential cross traffic that has an average rate of $\lambda = 12.5$, 25, and 37.5 Mbps, respectively. The third data set is generated for multiple tight links with the capacity $C = 100$ Mbps and exponential cross traffic with an average rate of $\lambda = 50$ Mbps. The fourth data set is for networks where the tight link is different from the bottleneck link, considering two scenarios: first, in which the tight link follows the bottleneck link and second, in which it precedes the latter. In both scenarios, the tight link capacity is $C = 100$ Mbps and the bottleneck capacity is $C_b = 50$ Mbps with cross traffic intensity of $\lambda = 75$ Mbps and $\lambda_b = 12.5$ Mbps, respectively.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] X. Liu, K. Ravindran, D. Loguinov, A queueing-theoretic foundation of available bandwidth estimation: Single-hop analysis, IEEE/ACM Trans. Netw. 15 (4) (2007) 918–931.

[2] M. Jain, C. Dovrolis, End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput, IEEE/ACM Trans. Netw. 11 (4) (2003) 537–549.

[3] B. Melander, M. Bjorkman, P. Gunningberg, A new end-to-end probing and analysis method for estimating bandwidth bottlenecks, in: IEEE Globecom, 2000, pp. 415–420.

[4] A. Johnsson, B. Melander, M. Björkman, Diettopp: A first implementation and evaluation of a simplified bandwidth measurement method, in: Swedish National Computer Networking Workshop, 2004.

[5] C. Dovrolis, P. Ramanathan, D. Moore, What do packet dispersion techniques measure?, in: IEEE INFOCOM, 2001, pp. 905–914.

[6] M. Jain, C. Dovrolis, Pathload: A measurement tool for end-to-end available bandwidth, in: Passive and Active Measurement Workshop, 2002.

[7] V.J. Ribeiro, R.H. Riedi, R.G. Baraniuk, J. Navratil, L. Cottrell, PathChirp: Efficient available bandwidth estimation for network paths, in: Passive and Active Measurement Workshop, 2003.

[8] J. Strauss, D. Katabi, F. Kaashoek, A measurement study of available bandwidth estimation tools, in: ACM Internet Measurement Conference, 2003, pp. 39–44.

[9] N. Hu, P. Steenkiste, Evaluation and characterization of available bandwidth probing techniques, IEEE J. Sel. Areas Commun. 21 (6) (2003) 879–894.

[10] S. Ekelin, M. Nilsson, E. Hartikainen, A. Johnsson, J.-E. Mangs, B. Melander, M. Bjorkman, Real-time measurement of end-to-end available bandwidth using Kalman filtering, in: IEEE/IFIP Network Operations and Management Symposium (NOMS), 2006, pp. 73–84.

[11] X. Liu, K. Ravindran, D. Loguinov, A stochastic foundation of available bandwidth estimation: Multi-hop analysis, IEEE/ACM Trans. Netw. 16 (1) (2008) 130–143.

[12] J. Liebeherr, M. Fidler, S. Valaee, A system theoretic approach to bandwidth estimation, IEEE/ACM Trans. Netw. 18 (4) (2010) 1040–1053.

[13] R. Lübben, M. Fidler, J. Liebeherr, Stochastic bandwidth estimation in networks with random service, IEEE/ACM Trans. Netw. 22 (2) (2014) 484–497.

[14] M. Jain, C. Dovrolis, Ten fallacies and pitfalls on end-to-end available bandwidth estimation, in: ACM Internet Measurement Conference, 2004, pp. 272–277.

[15] L. Lao, C. Dovrolis, M. Sanadidi, The probe gap model can underestimate the available bandwidth of multihop paths, ACM SIGCOMM Comput. Commun. Rev. 36 (5) (2006) 29–34.

[16] R. Prasad, M. Jain, C. Dovrolis, Effects of interrupt coalescence on network measurements, in: Passive and Active Measurement Workshop, 2004, pp. 247–256.

[17] Q. Yin, J. Kaur, Can machine learning benefit bandwidth estimation at ultra-high speeds? in: Passive and Active Measurement Conference, 2016, pp. 397–411.

[18] S. Keshav, A control-theoretic approach to flow control, in: Proc. ACM SIGCOMM, 1991, pp. 3–15.

[19] V. Paxson, End-to-end internet packet dynamics, IEEE/ACM Trans. Netw. 7 (3) (1999) 277–292.

[20] Z. Bozakov, M. Bredel, Online estimation of available bandwidth and fair share using Kalman filtering, in: IFIP Networking, 2009.

[21] A. Eswaradass, X.-H. Sun, M. Wu, A neural network based predictive mechanism for available bandwidth, in: Parallel and Distributed Processing Symposium, 2005.

[22] L.-J. Chen, A machine learning-based approach for estimating available bandwidth, in: TENCON, 2007, pp. 1–4.

[23] P. Hága, S. Laki, F. Tóth, I. Csabai, J. Stéger, G. Vattay, Neural network based available bandwidth estimation in the ETOMIC infrastructure. in: TRIDENTCOM, 2007, pp. 1–10.

[24] N. Sato, T. Oshiba, K. Nogami, A. Sawabe, K. Satoda, Experimental comparison of machine learning-based available bandwidth estimation methods over operational LTE networks, in: IEEE Symposium on Computers and Communications, ISCC, 2017, pp. 339–346.

[25] S. Khangura, F. Markus, B. Rosenhahn, Neural networks for measurement-based bandwidth estimation, in: IFIP Networking, 2018.

[26] D.S. Anderson, M. Hibler, L. Stoller, T. Stack, J. Lepreau, Automatic online validation of network configuration in the emulab network testbed, in: IEEE International Conference on Autonomic Computing, 2006, pp. 134–142.

[27] P. Emmerich, S. Gallenmüller, D. Raumer, F. Wohlfart, G. Carle, MoonGen: A Scriptable High-Speed Packet Generator, in: ACM Internet Measurement Conference, 2015.

[28] S. Avallone, S. Guadagno, D. Emma, A. Pescape, G. Ventre, D-ITG distributed internet traffic generator, in: Quantitative Evaluation of Systems, 2004, pp. 316–317.

[29] J. Laine, S. Saaristo, R. Prior, Real-time udp data emitter (rude) and collector for rude (crude), 2000, URL: https://sourceforge.net/projects/rude/.

[30] C.E. Rasmussen, C.K. Williams, Gaussian Processes for Machine Learning, vol. 38, The MIT Press, Cambridge, MA, USA, 2006, pp. 715–719.

[31] Bandwdith estimation traces for training and for evaluation, https://www.ikt.uni-hannover.de/bandwidthestimationtraces.html.